ulm university universität ulm

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme (DBIS)

Konzeption und Realisierung eines Konfigurators zur Erstellung webbasierter Fragebögen

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Maximilian Scheurich
maximilian.scheurich@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Michael Stach

2017

Fassung 12. Dezember 2017

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Die Verwendung von Fragebögen als Instrument zur Sammlung von Informationen ist in der Psychologie allgegenwärtig. Die Zunahme der Digitalisierung ist im Vergleich zu anderen Bereichen dennoch gering. Der Großteil der in der Psychologie verwendeten Fragebögen wird immer noch auf Papier gefertigt. Ursache hierfür kann zum einen fehlende IT-Kenntnis in diesen Bereichen und zum anderen der immense Kostenaufwand bei der Entwicklung von softwaregestützten Systemen sein. Darüber hinaus garantiert eine extern entwickelte Software keinen hundertprozentigen Erfolg. Das liegt vor allem an der fehlenden Fachkenntnis der Softwareentwickler. In den wenigsten Fällen ist der entwickelnde Softwarespezialist auch ein Fachexperte der Psychologie, was zu Fehlinterpretationen von Anforderungen in der Entwicklung der Software führen kann.

Diese Arbeit definiert Anforderungen auf der Grundlage gängiger Fragebögen, die es ermöglichen einen Fragebogen auf digitalem Wege zu entwickeln. Es werden relevante Stellen des Entwicklungsprozesses eines Fragebogens betrachtet, um den Entwicklungsprozesses zu demonstrieren. Ziel dieser Arbeit ist es, die fehlenden Fachkenntnisse der beiden Bereiche, Software-Entwicklung und Psychologie, mithilfe einer webbasierten Software auszugleichen. Es wird dafür ein Werkzeug entwickelt, das es gestattet die Komplexität einer Softwareentwicklung und die Erfassung aller Fragetypen einem ungeschulten Benutzer einfach zugänglich zu machen.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt und motiviert haben.

Ein besonderes Dankeschön geht an meinen Betreuer Michael Stach, der für hilfreiche Anregungen und konstruktive Kritik bei der Erstellung dieser Arbeit gesorgt hat.

Inhaltsverzeichnis

1 Einleitung

Folgende Abschnitte bringen dem Leser die Motivation dieser Arbeit, sowie die damit einhergehende Zielsetzung und auftretende Probleme näher. Zuletzt folgt eine Beschreibung des Aufbaus dieser Arbeit.

1.1 Motivation

Die Erfassung und Analyse von Daten spielt in der heutigen Zeit eine große Rolle. Sowohl im Bereich des Marketing, als auch in der Forschung werden große Mengen an Informationen gesammelt und ausgewertet. Während Branchen wie die Musik-, Film- oder Reiseindustrie bereits weitestgehend digitalisiert wurden, ist der Fortschritt der Digitalisierung im Gesundheitswesen, vor allem in Deutschland, als gering einzustufen [11].

Eine bewährte Methode zur Datenerhebung in der Forschung ist die Erstellung und Auswertung von Fragebögen. Mithilfe eines Fragebogens lassen sich leicht zuverlässige Aussagen einer Vielzahl an Personen untersuchen. Vor allem in der Psychologie sind Fragebögen das Mittel der Wahl.

Man unterscheidet im Allgemeinen drei Arten von Fragebogenformen: Erstens, die papiergebundene Form, bei der die Befragten mit Papier und Stift die Fragen selbständig beantworten. Zweitens, die digitale Form, die häufig in Online-Umfragen auftritt und in der Regel nur vordefinierte Antworten zur Auswahl lässt. Und zuletzt das Fragebogeninterview, bei dem ein Interviewer die Antworten zu den gestellten Fragen protokolliert. Dem Interviewer liegt hier in der Regel ein standardisierter Fragebogen zur Hand.

Ein großer Vorteil des Interviews ist die Möglichkeit zur Klärung inhaltlicher Fragen während des Prozesses. Diese Vorzüge bietet, bei einer ausreichend dynamischen Implementierung, auch ein digitalisierter Fragebogen. Aus einem ökonomischen Blickwinkel betrachtet, bietet die digitale Form und das Interview noch einen

weiteren Vorteil: Die Kostensenkung im Anwendungsfall. Größere Umfragen führen oftmals dazu, dass eine große Anzahl an Fragebögen benötigt wird und somit die Kosten für Papier und Druck steigen. Bei einem digitalisierten Fragebogen entfallen diese Kosten. Betrachtet man allerdings die Personalkosten für ein Interview oder die Entwicklungskosten für einen digitalen Fragebogen, dann ist ersichtlich, dass jede der drei Formen ihre Kehrseite besitzt. Das gilt auch für die Auswertung der Fragebögen, da auch bei einer computergesteuerten Auswertung zuvor Arbeit investiert werden muss.

1.2 Problemstellung

Die Semantik des Begriffes „Fragebogen“ ist breit gefächert. So kann man zwischen teilstandardisierten Fragebögen, die dem Befragten eine genau definierte Struktur vorgeben, aber auch freie Antworten erlauben und dem standardisierten Fragebogen unterscheiden. Letzterer beschränkt sich auf vordefinierte Antwortmöglichkeiten und lässt keinen Spielraum für individuelle Lösungen. Ein weiterer Typ von Fragebogen ist der normierte Fragebogen, welchem bereits umfangreiche Antworten als Vergleichskriterien zugrunde liegen und eine statistische Beurteilung erlaubt. In der Psychologie unterscheidet man noch zwischen Persönlichkeitsfragebögen und Intelligenztests. Bei der Bewertung von Intelligenztest kommt es allerdings zu keiner empirischen Bewertung, da die Antworten meistens nur wahr oder falsch sein können.

Wie in Abschnitt 1.1 angedeutet, bietet eine digitale Fragebogenform mehrere Vorteile im Anwendungsfall. Dennoch ist die Entwicklung eines digitalen Fragebogens für einen Computer-Laien kaum zu bewerkstelligen. Die Beauftragung eines Entwicklerstudios für eine solche Aufgabe sprengt häufig den finanziellen Rahmen und stellt somit keine zufriedenstellende Alternative dar.

Wie die Einführung der elektronischen Steuererklärung zeigte, wird der digitale Weg zu Anfang nicht von allen befürwortet [13]. Viele Personen bevorzugen noch den Papierweg. Der Grund hierfür ist oftmals die Gewohnheit oder die fehlende Kenntnis bei der Anwendung von Software. Obwohl die Tendenz sich in diesem Bereich nachhaltig ändert, ist dennoch deutlich zu erkennen, dass dadurch der Aufwand einer digitalen Fragebogenentwicklung im Vergleich zum Nutzen weiter steigt.

1.3 Zielsetzung

Ein häufig auftretendes Problem bei der Softwareentwicklung ist die fehlende Fachkenntnis der Entwickler über den adressierten Anwendungsbereich. Es kann daher nicht davon ausgegangen werden, dass jeder Softwareentwickler eine genaue Vorstellung davon hat wie ein Fragebogen aufgebaut ist, geschweige denn Grundkenntnisse der Psychologie besitzt.

Ziel dieser Arbeit ist es die Lücke zwischen den beiden Domänen der Psychologie und der Softwareentwicklung mithilfe eines Softwaresystems zu schließen. Es benötigt daher ein Werkzeug, das es gestattet die Komplexität einer Softwareentwicklung und die Erfassung aller Typen von Fragebögen einem ungeschulten Benutzer vereinfacht zugänglich zu machen. Die Verwendung eines solchen Werkzeugs sollte es dem Benutzer erlauben alle Bereiche der Fragebogenentwicklung steuern zu können, ohne ihn in deren Umfang einzuschränken. Dadurch verringert sich der Arbeits- und Kostenaufwand für einen digitalen Fragebogen erheblich, was die Chancen einer vollständigen Digitalisierung erhöht.

Die Anwendung soll geräteunabhängig, sowie größtenteils softwareunabhängig ausgeführt werden können. Es sollte jedem Benutzer ermöglicht werden mit geringem Aufwand einen digitalen Fragebogen zu erstellen. Das dafür zu entwickelnde System soll eine anwenderfreundliche Benutzerschnittstelle bieten und im weitesten Sinne selbsterklärend sein.

1.4 Struktur der Arbeit

Die Arbeit ist wie folgt in sieben Kapitel unterteilt. In **Kapitel 2** werden mehrere verwandte Arbeiten vorgestellt. Es werden die Kernaspekte dieser Arbeiten erläutert, um eine Einordnung des zu entwickelnden Systems vorzunehmen. Das **Kapitel 3** befasst sich ausschließlich mit den Anforderungen an das System. Sowohl funktionale, als auch nicht-funktionale Anforderungen werden in diesem Abschnitt definiert. Das **Kapitel 4** soll einen Einblick in den Aufbau des Systems geben und wird durch das folgende **Kapitel 5** ergänzt, in dem die wichtigsten verwendeten Technologien vorgestellt werden. **Kapitel 6** nimmt Bezug auf die vorher definierten Anforderungen und geht auf ihre Umsetzung ein. In **Kapitel 7** wird abschließend ein Fazit über das behandelte Thema gezogen.

2 Verwandte Arbeiten

Dieses Kapitel setzt sich mit verwandten Arbeiten und deren Umsetzung auseinander. Es soll den aktuellen Stand des behandelten Themas widerspiegeln, sowie erste Erkenntnisse bezüglich der Anforderungen und Problemstellung liefern.

2.1 QuestionSys

QuestionSys [8] ist ein Projekt des Instituts für Datenbanken und Informationssysteme der Universität Ulm, welches sich mit der Erstellung und Auswertung von mobilen Fragebögen im Bereich der Psychologie befasst. Ziel dieses Projektes ist die Minimierung der Arbeit, die bei der Entwicklung von komplexen Auswertungsbögen anfällt. Die Anwendung des Systems bezieht sich hauptsächlich auf die Benutzung mobiler Endgeräte. Wie die Abbildung 2.1 verdeutlichen soll, ermöglicht QuestionSys eine übersichtliche und einfach zu handhabende Darstellung eines komplexen Arbeitsschritts bei der Erstellung von geführten Fragebögen.

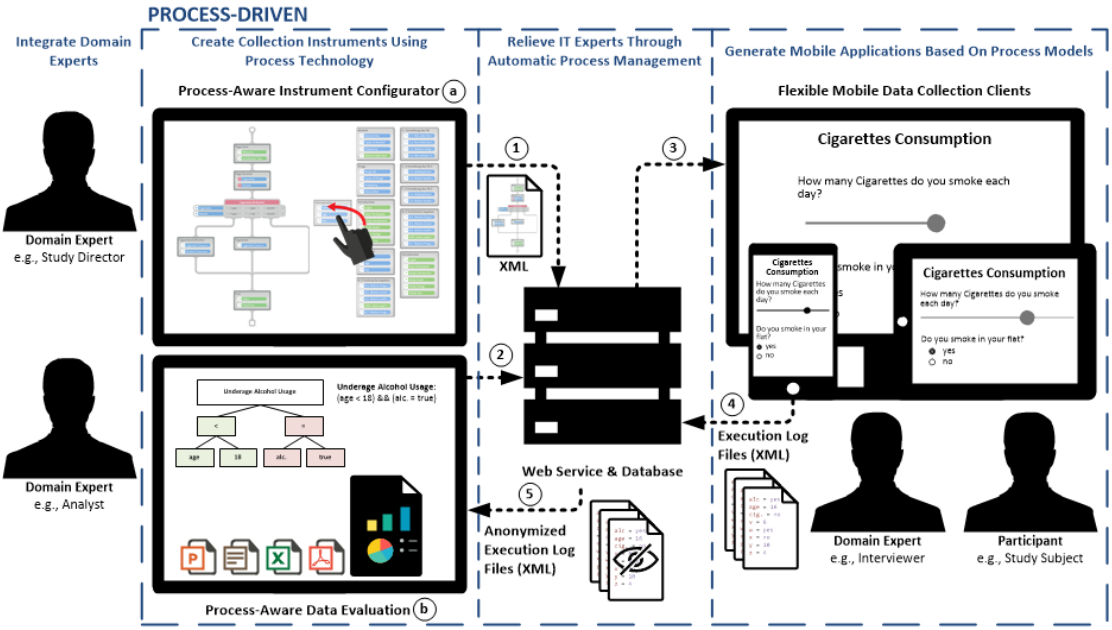

Abbildung 2.1: Konzept für eine prozessgetriebene mobile Datenerfassung [18]

Die Erstellung eines Fragebogens ist in Form von Prozessmodellen definiert. Die Bereitstellung eines Editors ⓐ für eine prozessorientierte Modellierung eines Fragebogens erlaubt dem Fachexperten eine einfache Verwendung. Darüber hinaus unterstützt QuestionSys den gesamten Lebenszyklus eines digitalen Fragebogens, bis hin zur Evaluierung durch einen Fachexperten. Dieser wird ebenfalls in seiner Aufgabe durch eine Datenaufbereitung ⓑ unterstützt. Diese Arbeit konnte zeigen, dass elektronische Fragebögen den Fachexperten von kostspieligen manuellen Aufgaben entlasten, wie die Übertragung, Transformation und Analyse der gesammelten Daten [20, 17, 19].

2.2 MobileTx

Das MobileTx-Projekt [5] des Instituts für Datenbanken und Informationssysteme der Universität Ulm ist darauf bedacht therapeutischen Leistungen in das Leben eines Patienten zu übertragen. Dies soll vor allem den therapeutischen Prozess fördern und den Therapeuten in seiner Position entlasten. Unter Verwendung des Konzepts der „Hausaufgaben“ wird der Fortschritt eines Patienten gemessen. Der

Therapeut hat die Möglichkeit Aufgaben an seine Patienten zu verteilen. Das unterstützt den therapeutische Prozess auch außerhalb der Sitzungen und gibt dem Therapeuten gleichzeitig ein Feedback, was zu einer besseren Beurteilung des Patienten führen kann. Dabei sind vor allem vier Hauptaspekte zu berücksichtigen: (1) die Erstellung von Übungen, (2) die Spezifikation eines Kontextes, (3) die Bereitstellung eines Feedback-Features und (4) eine Benachrichtigungsfunktion. Zusammen bieten diese vier Aspekte einen Ansatz, der therapeutische Interventionen mit dem Einsatz von mobilen Geräten unterstützt und zunehmen wichtiger wird [16]. Dieser Vorgang kann nicht nur in der Psychotherapie, sondern auch in anderen Behandlungen nützlich sein und die Kommunikation zwischen Therapeuten und Patienten verbessern.

Das Projekt soll die heutige Smartphonetechnologie mit prozessorientierten Informationssystemen und klinischer Erfahrung kombinieren. Die daraus hervorgehenden E-Health-Tools lassen sich so anhand deren Wirksamkeit empirisch bewerten.

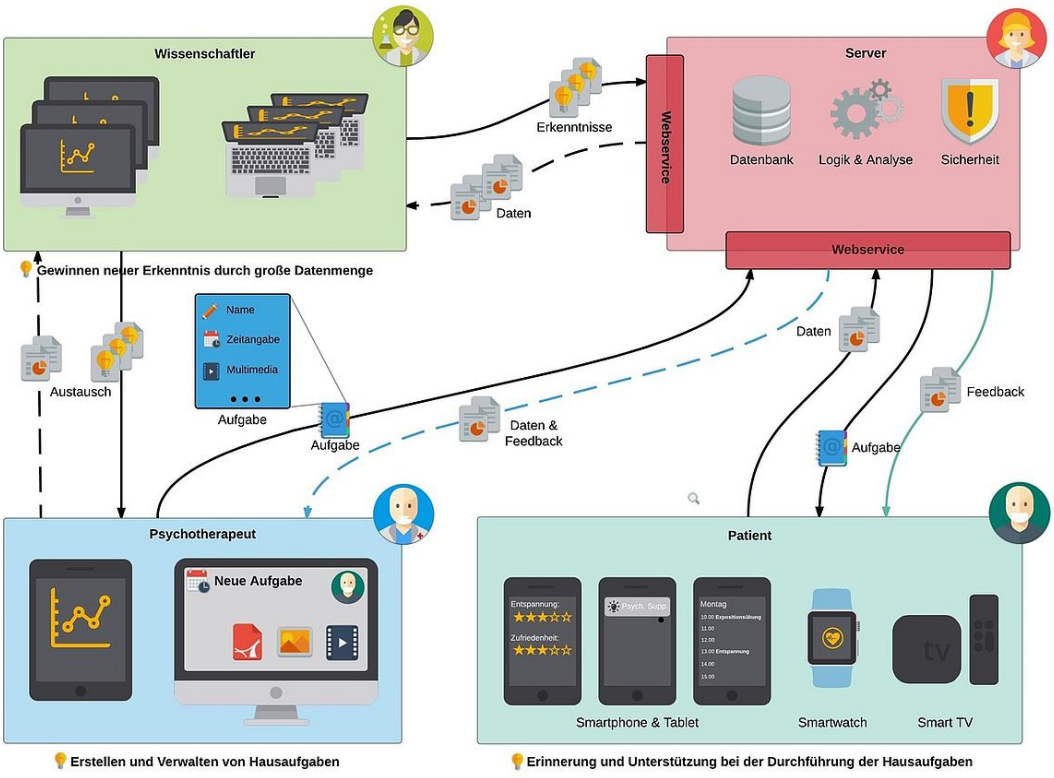

Abbildung 2.2: Aktivitätsdiagramm für den Einsatz von MobileTx [5]

Die Abbildung 2.2 zeigt die Verwendung von MobileTx unter Einbindung der be-

teiligten Akteure. Unter der Rubrik des Patienten ist außerdem zu erkennen, dass ein Feedback hauptsächlich durch die Verwendung von Fragebögen erzielt wird. An dieser Stelle bietet sich die Möglichkeit eines ersten Anwendungsfalles für die Implementierung, die im Rahmen dieser Arbeit entwickelt wird, an. Durch diese Erkenntnis wird deutlich, dass eine der Anforderungen an das System Bezug auf mobile Endgeräte nehmen muss.

2.3 LimeSurvey

LimeSurvey [10] ist ein Open-Source-Projekt der LimeSurvey GmbH, das es erlaubt detaillierte Online-Umfragen ohne Programmierkenntnisse zu erstellen. Die erstellte Umfrage lässt sich direkt über LimeSurvey veröffentlichen. Eine Einbindung in externe Systeme ist bedingt über diverse Integrationsmodule möglich. Alle Ergebnisse einer Umfrage werden nach Abschluss in einer Datenbank erfasst. Layout und Design der Umfrage sind durch ein Vorlagensystem veränderbar.

Neben einfachen Fragetypen, wie Ja-/Nein-Fragen, lassen sich in LimeSurvey auch Mehrfachantwortfragen einbinden. Matrixmodelle und Gewichtungsfragen, sowie Fragetypen mit freier Antwortmöglichkeit stehen ebenfalls zur Auswahl. Über eine Regelfunktion lassen sich Gruppen bilden und Filterfragen erstellen. Bilder und Videos können ganz ebenfalls einfach durch das Anklicken einer Schaltfläche hinzugefügt werden.

Ein Bedienung von LimeSurvey ohne Bedienungsanleitung gestaltet sich aufgrund der Vielzahl an Funktionalitäten schwierig. Bei der Erstellung von Fragebögen setzt LimeSurvey auf eine hierarchische Struktur, bei der auf der untersten Ebene die einzelnen Fragen bearbeitet werden können.

Während einer Umfrage bietet LimeSurvey die Möglichkeit aktuellen Ergebnisse einzusehen. Zur Veranschaulichung der Ergebnisse, können diese sogar in Form von Diagrammen visualisiert werden. Die nachfolgende Abbildung 2.3 zeigt eine mögliche Umfrage, die mit LimeSurvey erstellt wurde.

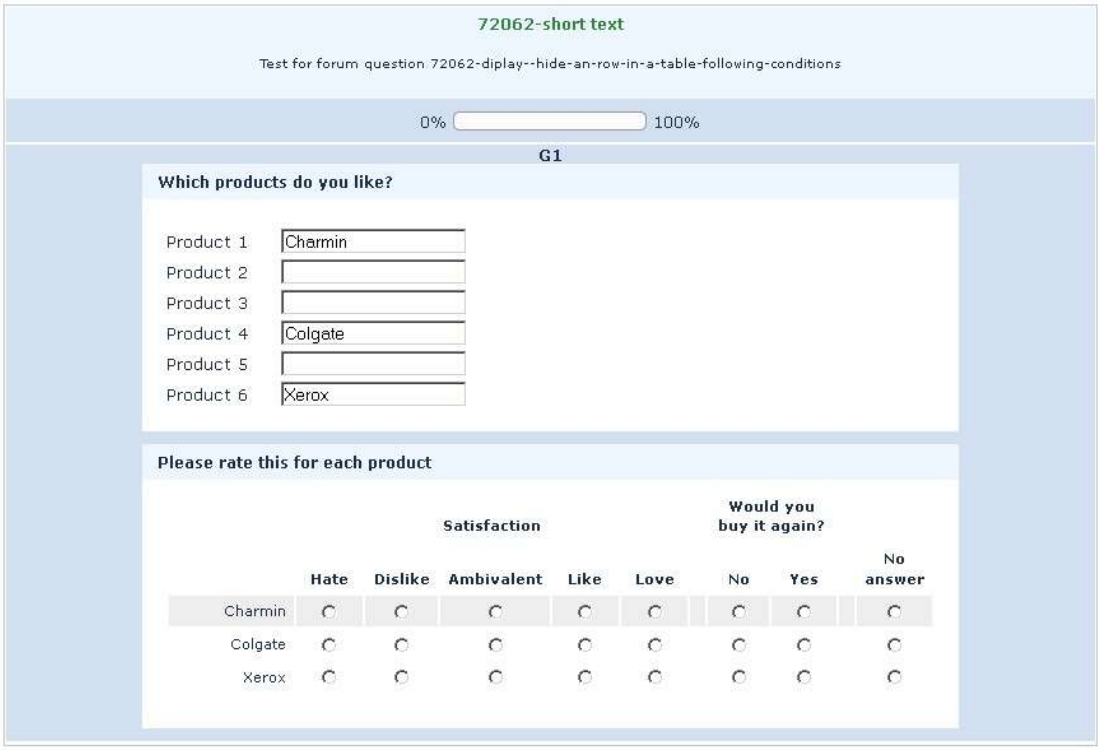

Abbildung 2.3: Demo einer LimeSurvey Umfrage [10]

3 Anforderungsanalyse

In diesem Kapitel werden anhand der gegebenen Problemstellung Anforderungen an das System definiert. Es wird zunächst eine Benutzerprofilanalyse erstellt, um den Umfang der Anforderungen genauer einordnen zu können. In Abschnitt 3.2 wird bei der Systemaufgabenanalyse zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden.

3.1 Aufgabenanalyse

In diesem Abschnitt werden die genauen Aufgaben des zu implementierenden Systems anhand von Benutzerprofilen und simulierten Szenarien eingegrenzt. Dieser Prozess spielt vor allem bei der Entwicklung von Anwendungssystemen eine große Rolle. Je detaillierter die Anforderungen an das System herausgearbeitet werden, desto wahrscheinlicher ist es funktionsfähige und brauchbare Software herzustellen, da mögliche Engpässe frühzeitig erkannt werden können.

3.1.1 Einsatzziele

Die Einsatzziele von Fragebögen sind über alle Bereiche des Lebens verteilt. Man begegnet ihnen bei Online-Umfragen, sowie bei den örtlichen Behörden in Form von Formularen. In anderen Bereichen, wie zum Beispiel der Psychologie ist die Absicht der Verwendung von Fragebögen offensichtlicher. Unabhängig von der Art des Auftretens erfüllen Fragebögen im Allgemeinen den Zweck, Informationen und Meinungen bezüglich eines im Fragebogen angesprochenen Themas zu sammeln. Die Auswertung führt häufig zu einem besseren Verständnis der Verhaltensweise von Personen in bestimmten Situationen oder zur Aufnahme von Personendaten ohne analytischen Hintergrund.

Insbesondere bei der Entwicklung von Fragebögen im Bereich der Psychologie kann es zu hohem Arbeitsaufwand kommen. Der Grund hierfür ist der analytische

Aspekt, welcher über eine gezielte Fragestellung erreicht werden soll. Bei der Beantragung eines Führerscheins zum Beispiel sind die gestellten Fragen unabhängig von der Personengruppe immer gleich aufgestellt. Eine psychologische Umfrage bezüglich des Trinkverhaltens hingegen muss zwischen Personengruppen differenziert und teilweise nach Antwortmöglichkeiten gefiltert werden. Ein hervorragende Einsatzmöglichkeit von digitalen Fragebögen liefert der Artikel „Outpatient Tinnitus Clinic, Self-Help Web Platform, or Mobile Application to Recruit Tinnitus Study Samples?“ [14], in welchem die Sammlung von Informationen zum Krankheitsbild „Tinnitus“ beschrieben wird.

Besonders in der Psychologie würde man von einer verbesserten Entwicklungsmethode profitiert. Somit fokussiert sich das Einsatzziel, welches im Rahmen dieser Arbeit betrachtet wird, auf die Entwicklung psychologischer Fragebögen.

3.1.2 Benutzerprofilanalyse

Softwareentwicklung, insbesondere die Entwicklung von Benutzerschnittstellen ist abhängig von der jeweiligen Benutzergruppe. Da bei einer homogenen Gruppe von Nutzern, welche sich in relevanten Eigenschaften ähneln, auch ähnliche Erwartungen an Benutzerschnittstellen vorausgesetzt werden können, fasst man sie in einer Benutzerkategorie zusammen. Für jede Benutzerkategorie wird ein Benutzerprofil erstellt, welches Aufschluss über die Entscheidung im Hinblick einer Benutzerschnittstelle liefern kann. Unter der Prämisse, dass sich die Benutzergruppe hauptsächlich auf den Bereich der Forschung, insbesondere der Psychologie, beschränkt, wird ein Psychologe die Stelle des Repräsentanten einnehmen.

Die Erwartungen und Kenntnisse dieses Benutzerprofils können wie folgt beschrieben werden. Ein Psychologe besitzt in der Regel umfangreiche Kenntnisse im Bereich der Psychologie. Aufgrund seiner Ausbildung beschränken sich die IT-Kenntnisse häufig nur auf die Verwendung von Anwendungen. Im Bereich der Anwendungssoftware wird eine simple und intuitive Bedienung vorausgesetzt, die effektives Arbeiten ermöglicht. Dennoch kann davon ausgegangen werden, dass Funktionalität vor Design gestellt wird. Diese Beurteilung schließt sich aus dem Berufsprofil, welches als rational und strukturiert beschrieben wird.

Basierend auf den relevanten Eigenschaften eines Psychologen lässt sich schließen, dass jede Person als Benutzer in Frage kommt, welche ein Grundverständnis von Anwendungssoftware besitzt, keine Ausbildung in der Softwareentwicklung absolviert hat und mit der Erstellung von Fragebögen vertraut ist.

3.1.3 Szenarien

In diesem Abschnitt werden zwei hypothetische Szenarien vorgestellt, welche die Probleme einer papiergebundenen Fragebogen-Entwicklung und -Anwendung verdeutlichen.

Bei größeren Studien in der Psychologie kann es häufiger vorkommen, dass Fragebögen an gewisse Personengruppen angepasst werden müssen oder Filterfragen auf einen anderen Abschnitt verweisen. Im Falle einer papiergebundenen Vorlage eines Fragebogens müssen die Verantwortlichen für die korrekte Ausfüllung des richtigen Fragebogenabschnitts sorgen, um später valide Ergebnisse zu erhalten.

Ein weiteres Problem der papiergebundenen Anwendung von Fragebögen ist der organisatorische Aufwand, der in vielen Fällen betrieben werden muss. Angenommen ein Institut möchte eine Umfrage in einem nicht regionalen Umfeld durchführen und hat keine Möglichkeit am Umfrageort die entwickelten Fragebögen auszudrucken, dann bleibt letztendlich nur die Möglichkeit eine autorisierte Gruppe zusammen mit den ausgedruckten Fragebögen an den gewünschten Ort zu schicken. Dieser Vorgang kostet nicht nur Geld sondern auch Zeit und verlangsamt ebenfalls den Vorgang der Auswertung, da die Ergebnisse zuvor wieder das Institut erreichen müssen. Kommt es jetzt noch zu außerplanmäßigen Zwischenfällen, die ein Anpassen der Fragebögen erfordern wird dieser Vorgang nahezu unmöglich.

3.2 Systemaufgaben

Der folgende Abschnitt beschäftigt sich mit den Aufgaben, die an das System gestellt werden. Es werden alle zuvor angesprochenen Funktionen in konkrete Anforderungen formuliert.
Die Priorisierung der Anforderungen wird im Sinne der MoSCoW-Priorisierung [6] durchgeführt. Die MoSCoW-Priorisierung ist eine Methode, die im Bereich des Projektmanagements erfolgreich zur Priorisierung der Anforderungsumsetzung anhand ihrer Wichtigkeit eingesetzt wird.

Nach MoSCoW wird folgender Wertebereich differenziert:

- **MUSS** (eng.: MUST, unbedingt erforderlich)
- **SOLL** (eng.: SHOULD, sollte umgesetzt werden)
- **KANN** (eng.: COULD, kann umgesetzt werden, sollten wichtigere Anforderungen schon abgeschlossen sein)
- **SPÄTER** (eng.: WONT, wird für einen späteren Zeitpunkt vorgemerkt)

3.2.1 Funktionale Anforderungen

In diesem Abschnitt geht es um die konkreten funktionalen Aufgaben, die das System erfüllen soll. Die aufgelisteten Anforderungen werden in nachfolgenden Tabellen priorisiert und beschrieben.

Allgemeine funktionale Anforderungen

Es werden grundsätzliche Anforderungen, die nicht direkt mit der Fragebogenthematik zusammen hängen aufgelistet.

BEZEICHNUNG	PRIORITÄT	BESCHREIBUNG
Speichern	*MUSS*	Der Benutzer muss aktuelle Stände seiner Fragebögen im Browserspeicher für eine spätere Bearbeitung zwischenlagern.

Laden	*MUSS*	Der Benutzer muss abgelegte Fragebögen aus dem lokalen Browserspeicher zur weiteren Bearbeitung abrufen können.
Löschen	*MUSS*	Der Benutzer muss den aktuell ausgewählten Fragebogen löschen können.
Externe Anbindung	*MUSS*	Der Benutzer muss die Möglichkeit haben den fertigen Fragebogen auf einer externen Plattform bereitzustellen.
Import	*SOLL*	Der Benutzer soll bereits erstellte Fragebögen oder Vorlagen des hier verwendeten Datentyps zur weiteren Verarbeitung importieren können.
Export	*SOLL*	Der Benutzer soll Fragebögen exportieren können. Diese Funktion soll zu Speicherzwecken oder auch zur weiteren Verwendung des Fragebogens in einem externen System benutzt werden können.
Drag & Drop	*SOLL*	Der Benutzer soll über eine Drag & Drop Funktion neue Fragebogenelemente an den aktuellen Fragebogen hinzuzufügen können. Diese Funktion gestattet es u.a. mobilen Endgeräten, die über eine Touch-Screen-Eingabe verfügen, mit dem System zu interagieren.
Preview	*SOLL*	Der Benutzer soll sich über eine Preview-Funktion den aktuellen Stand seines Fragebogens im Anwendungsfall anzeigen und dessen Ergebnisse einsehen können.
Direkte Eingabe	*SOLL*	Der Benutzer soll das Datenobjekt, welches den Fragebogen definiert direkt bearbeiten können.
Metadaten	*SOLL*	Der Benutzer soll persönliche Daten, sowie ein Erstellungsdatum an seine Fragebogendatei anhängen können. Das bietet die Möglichkeit einer besseren Datenverwaltung bei steigender Fragebogenanzahl.

Validierung	*SPÄTER*	Der Benutzer hat die Möglichkeit sein Datenobjekt auf Korrektheit zu überprüfen.

Funktionale Anforderungen in Bezug auf die Fragebogenelemente

Es werden nachfolgend Anforderungen definiert, die sich direkt auf die verschiedenen Typen von Fragebogenelemente beziehen.

BEZEICHNUNG	PRIORITÄT	BESCHREIBUNG
	Umsetzung analoger Fragebogenelemente	
Texteingabe	*MUSS*	Der Benutzer muss Fragebogenelemente erstellen können, die eine textbasierte Antwort zulassen.
Zahleneingabe	*MUSS*	Der Benutzer muss Fragebogenelemente erstellen können, die eine zahlenbasierte Antwort zulassen.
Zeit-/Datumseingabe	*MUSS*	Der Benutzer muss Fragebogenelemente erstellen können, die eine Antwort in Datumsformat zulassen.
Multiplechoice-fragen	*MUSS*	Der Benutzer muss Fragebogenelemente erstellen können, welche Multiplechoice-Antworten zulassen.
Gewichtungs-eingaben	*MUSS*	Der Benutzer muss Gewichtungsfragen erstellen können.
Anmerkungen	*MUSS*	Der Benutzer muss Platz für optionale Kommentare anlegen können.
Informationstexte	*MUSS*	Der Benutzer muss informelle Textpassagen anlegen können.
Gruppierung	*SOLL*	Der Benutzer soll Elemente gruppieren können.
Seitenumbruch	*SOLL*	Der Benutzer soll einen Seitenumbruch einfügen können.

Bilderanzeige	*SOLL*	Der Benutzer soll Bilder einfügen können. Diese können als Referenz dienen.

Erweiterung der Fragebogenelemente

Filterfragen	*MUSS*	Der Benutzer muss abhängig von Antworten neue Abschnitte für den Befragten zuschalten können.
Dropdown-Auswahl	*MUSS*	Der Benutzer muss Fragebogenelemente erstellen können, die Antworten aus einer Dropdown-Auswahl zulassen.
Matrix-Auswahl	*MUSS*	Der Benutzer muss Fragebogenelemente erstellen können, die Antworten innerhalb einer Matrix zulassen. Die einzelnen Antworten können aus oben aufgelisteten Elementen bestehen.
Dateiupload	*SOLL*	Der Benutzer soll einen Bereich definieren können, in dem Dateien angehängt werden können. Dies bietet die Möglichkeit einer interaktiven Aufgabe, wie der Bearbeitung eines Bildes.
Bewegte Bilderanzeige	*KANN*	Es können Videos oder GIFs eingespielt werden.

3.2.2 Nicht-funktionale Anforderungen

In diesem Abschnitt werden die nicht-funktionalen Anforderungen an das System definiert. Diese spezifizieren im Wesentlichen die Qualitätsansprüche an Systemfunktionen oder das System als Ganzes. Die aufgelisteten Anforderungen werden in nachfolgenden Tabellen priorisiert und beschrieben.

BEZEICHNUNG	PRIORITÄT	BESCHREIBUNG
Benutzbarkeit	*MUSS*	Das System muss selbsterklärend und übersichtlich sein.
Zuverlässigkeit	*MUSS*	Das System muss zuverlässig funktionieren und verursacht keine unerwarteten Fehler.
Portierbarkeit	*MUSS*	Das System muss auf unterschiedlichen Plattformen zugänglich sein.
Korrektheit	*MUSS*	Die Ergebnisse, die das System liefert müssen fehlerfrei sein.
Aussehen und Handhabung	*SOLL*	Das System soll sich an allgemein akzeptierte Designkonventionen halten.
Leistung und Effizienz	*SOLL*	Das System soll schnell agieren und wenig Ressourcen verwenden.
Sicherheitsanforderungen	*SOLL*	Das System soll Daten vertraulich behandeln und sorgt für eine ständige Verfügbarkeit.
Flexibilität	*KANN*	Das System kann allgemeine Standards unterstützen.
Erweiterbarkeit	*KANN*	Das System kann sich mit wenig Aufwand erweitern oder ändern lassen.

4 Architektur

In diesem Kapitel wird die Architektur des Fragebogeneditors im Rahmen der definierten Anforderungen entworfen.

4.1 Allgemeiner Aufbau

Für eine bestmögliche Benutzbarkeit des Systems wird, im Rahmen dieser Arbeit, die Implementierung webbasiert umgesetzt. Des Weiteren gewährleistet eine webbasierte Anwendung durch die Plattformunabhängigkeit eine große Erreichbarkeit. Der Benutzer ist in diesem Fall nicht auf eine besondere Hardware oder ein spezielles Betriebssystem angewiesen.

Aufgrund des wiederholten Vorkommens einzelner Fragebogenelemente, die zuvor in Kapitel 3.2.1 definiert wurden, bietet sich eine leicht veränderte, wie in der Objektorientierten Programmierung verwendete, Model-View-Controller Struktur an. Der Aufbau des Systems gliedert sich folglich in drei Komponenten, wie Abbildung 4.1 zeigt. Die Darstellung wird bedingt durch die ausführende Plattform von einer HTML Datei übernommen. Als eine Art Controller kommt eine Renderinstanz des Frameworks VueJs zum Einsatz, welche das dynamische Rendern von JSON-Objekten erlaubt. In folgendem Kapitel 5.1.2 wird die Funktionsweise des Controllers, hier „ModelView“ genannt, genauer erklärt. Zuletzt werden mit der Skriptsprache JavaScript objektorientierte Klassen simuliert. Die nachfolgende Abbildung 4.1 dient der Verdeutlichung dieses Ansatzes und wird in den darauffolgenden Abschnitten weiter erläutert.

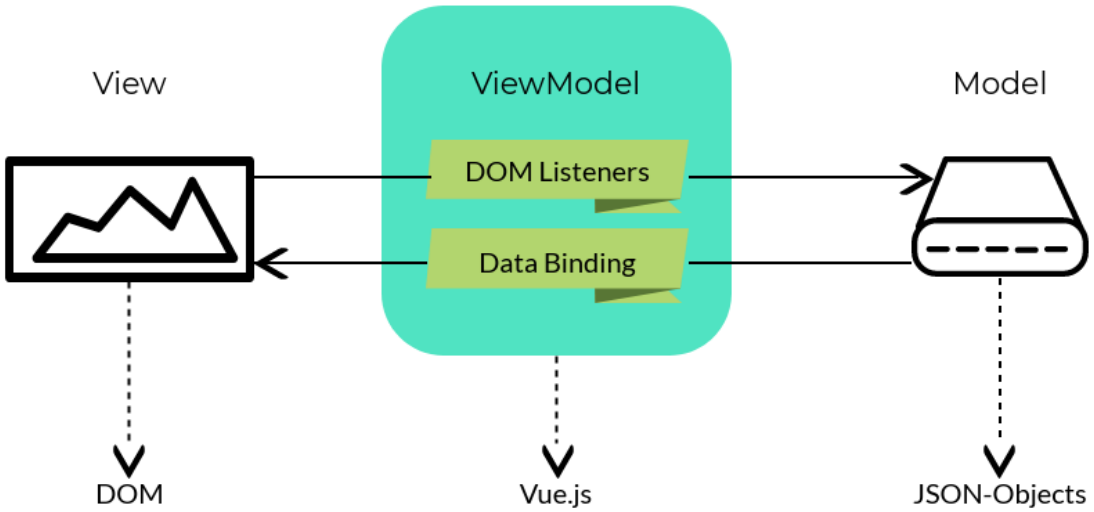

Abbildung 4.1: Grafische Darstellung der Model View ViewModel Struktur

4.2 Komponenten

In diesem Kapitel werden die angeführten Bereiche aus dem Abschnitt 4.1, in welche sich das System unterteilt, genauer erläutert. Es wird verdeutlicht, welche Funktion die jeweilige Komponente einnimmt und welche Vorteile sich dadurch bieten.

4.2.1 Darstellung

Die Darstellung der Anwendung, die im Rahmen dieser Arbeit entwickelt wird, soll mittels HTML realisiert werden. Dieser sogenannte Einzelseiten-Ansatz, auch „single-page application“ (kurz: SPA) genannt, reduziert die Kommunikation zwischen Server und Client und steigert somit die Effizienz. Obwohl es sich bei dieser Anwendung, um eine Frontend-Webanwendung handelt, bietet dieser Ansatz dennoch strukturelle Vorteile. Durch die Verwendung von nur einer HTML-Datei verringert sich die Anzahl der aufgebauten Verbindungen zum Server. Die Anwendung kann sogar weiterhin auf dem Client betrieben werden, ohne dass eine Verbindung zum Server bestehen muss, sobald alle nötigen Daten übertragen wurden.

4.2.2 Steuerung

Die Steuerung der Webanwendung wird von einer Renderinstanz übernommen, deren Funktionsweise in Kapitel 5.1.2 erklärt wird. Der Datenbindungsmechanismus der Instanz ermöglicht es die Fragebogenelemente in Form von JSON-Objekten, wie Listing 5.2 in nachfolgendem Kapitel zeigt, an die HTML-DOM (Dokumenten-Objekt-Modell) [2] anzubinden. Das Resultat ist eine grafische Darstellung der Fragebogenelemente, ähnlich der einer papiergebundenen Ausführung.

Über eine „Drag & Drop“ Funktion lassen sich auf diese Weise beliebig viele Elemente aus dem Bereich ① der Abbildung 4.2 aneinander ketten. Der aktuelle Fragebogen besteht somit aus einer Verkettung von JSON-Objekten, die als ein Datenobjekt an die Renderinstanz gebunden ist. Änderungen einzelner Elemente, wie zum Beispiel die Anpassung der Fragestellung, werden in Echtzeit von der Renderinstanz über eine Änderung innerhalb eines Formulars an das angebundene Datenobjekt übergeben und direkt in der HTML-DOM sichtbar gemacht. In den Bereichen ② und ③ der Abbildung 4.2 lässt sich dieser Prozess erkennen.

Es lassen sich natürlich auch Änderungen am gesamten Datenobjekt, welches alle Elemente beinhaltet, vornehmen. Diese Funktion ermöglicht es ganze Abschnitte zu kopieren, zu ändern, einzufügen oder zu löschen.

4.2.3 Modell

Wie in Abschnitt 4.1 beschrieben kann man in JavaScript mithilfe eines Funktionsaufrufes eine Klasse simulieren. Listing 4.1 soll dies verdeutlichen.

```
1  function Checkbox(count) {
2
3    this.type = "checkbox";
4    this.title = '';
5    this.choices = [
6      {
7        value: 'item1',
8        text: ""
9      },
10     {
11       value: 'item2',
12       text: ""
13     },
14     {
15       value: 'item3',
16       text: ""
17     }
18   ];
19   this.choicesOrder = "";
20   this.colCount = 1;
21   this.isRequired = false;
22   this.name = "question"+count;
23   this.width = "100%";
24   this.visibleIf = "";
25
26 }
```

Listing 4.1: Checkbox Klasse

Mit dieser Hilfsklasse lassen sich JSON-Objekte erstellen, wie der Aufruf Zeile 1 des Listings 4.2 zeigt.

```
1  var checkbox = new Checkbox(0);
2
3  checkbox.title = "new title";
4  var type = checkbox.type;
```

Listing 4.2: Aufruf eines Checkbox-Objekts

Die Attribute dieses Objekts können, wie Zeile 2f. angibt, angesprochen werden. Die einzelnen Formularobjekte werden bei ihrer Erstellung an die Renderinstanz gebunden. Wird eine Veränderung von der Renderinstanz registriert, dann werden

die beteiligten Komponenten neu gerendert. Die hierbei existierende Verbindung ist bidirektional gerichtet. Das bedeutet, dass eine Änderung über das User-Interface eine Änderung des zugrundeliegenden Objektes verursacht. Das Resultat ist in Abbildung 4.2 dargestellt. Das Datenobjekt lässt sich somit nicht nur in Bereich ② anzeigen, sondern direkt über das Formular in Bereich ③ verändern.

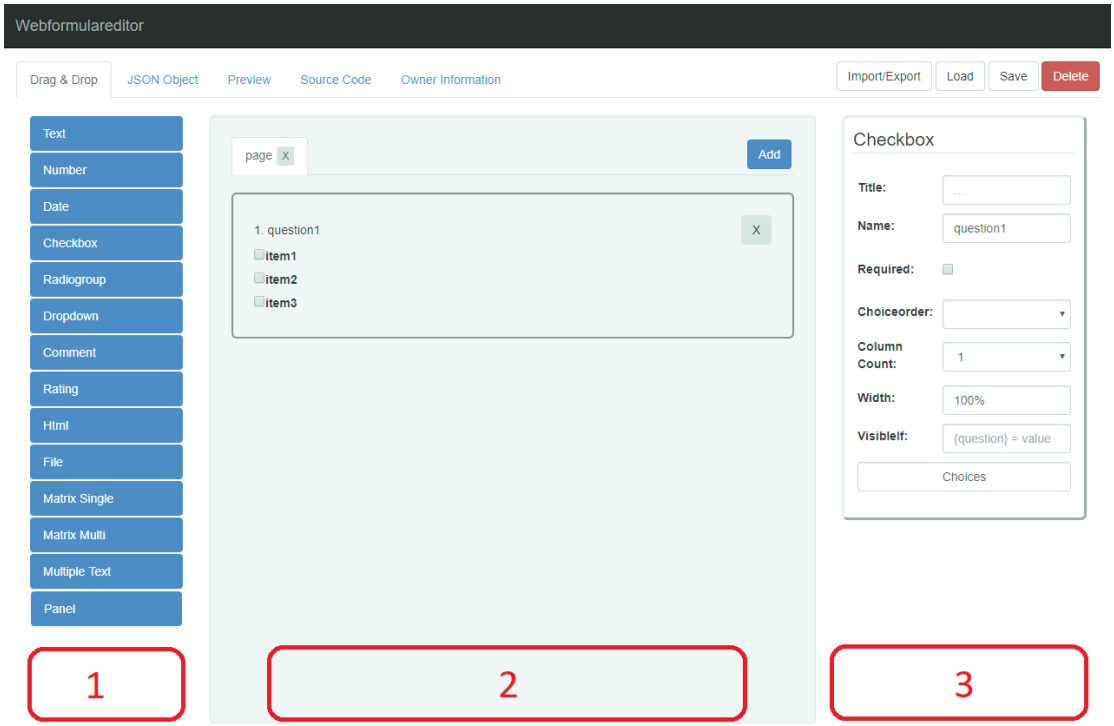

Abbildung 4.2: Ausschnitt eines gerenderten JSON-Objekts der Klasse Checkbox

Über das Formular, welches in Bereich ③ der Abbildung 4.2 zu sehen ist, werden Änderungen in Echtzeit an das Datenobjekt übergeben. Dieser Vorgang lässt sich auf alle Fragebogenelemente, die in Abschnitt 3.2.1 definiert wurden, übertragen.

5 Ausgewählte Implementierungsaspekte

In diesem Kapitel werden zuerst relevante Technologien vorgestellt und ihre Funktionen näher erläutert. Sie bilden die Grundlage für die darauffolgenden Implementierungsaspekte. Der Abschnitt 5.2 beschreibt den Einsatz von generischem Code als Mittel zur Vervielfältigung von Fragebogenelementen. Im letzten Abschnitt wird auf die Umsetzung der Benutzerschnittstelle eingegangen, die eine Wichtige Rolle bei der Erfüllung der aufgestellten Anforderungen spielt.

5.1 Verwendete Technologien

Als Grundlage für das System dient ein HTML-Dokument. Durch die Verwendung von JavaScript lassen sich weitere Inhalte dynamisch nachladen. In den folgenden Abschnitten werden die verwendeten Technologien vorgestellt und der resultierende Vorteil verdeutlicht.

5.1.1 SurveyJs

SurveyJs [7] ist ein Projekt der Firma Devsoft Baltic OU und befasst sich hauptsächlich mit der dynamischen Erzeugung von digitalen Fragebögen. Im Gegensatz zu anderen Lösungen im Bereich der Fragebogenverwaltung besteht das Projekt SurveyJs aus drei unabhängigen Teilen: Der Laufzeit-SurveyJs-Bibliothek, dem SurveyJs-Editor und dem SurveyJs-Service zum Speichern und Analysieren von Ergebnissen. Dem Benutzer soll so die Verwendung eines ganzen Systems, als auch die Integration eines Teilsystems, wie der Bibliothek, ermöglicht werden. Es kann der SurveyJs-Dienst verwendet werden, um die Ergebnisse dort oder in einer

eigenen Anwendung zu speichern. Die Entwickler stellen auf ihrer Seite die Open-Source SurveyJs-Bibliothek zur Verfügung, welche im allgemeinen für die Auswertung von Fragebögen auf Basis von JSON-Objekten zuständig ist.

Die SurveyJs-Bibliothek liefert, wie in Abschnitt 5.1.2 erklärt wird, eine Renderfunktionalität, die es erlaubt ein JSON-Objekt an einen HTML-Bereich zu binden. Die verschiedenen Attribute des Objekts werden je nach Wert in einen HTML Kontext gebracht und in den vorgesehenen Bereich eingebunden. Im Listing 5.1 wird anhand eines Beispiels die Einbindung der SurveyJs-Bibliothek erklärt.

```
1  Survey.Survey.cssType = "bootstrap";
2
3  try {
4    var surveyJSON = var surveyJSON = {
5      "pages":[{ "name":"page",
6             "elements":[{"type":"checkbox",
7                "title":"Wie viel Bier trinken sie am Tag?",
8                "choices":[{"value":"item1", "text":"keins"},
9                  {"value":"item2", "text":"eins"},
10                 {"value":"item3","text":"mehr als eins"}
11               ], "choicesOrder":"asc", "colCount":"1",
12               "isRequired":false, "name":"question1",
13                "width":"100%", "visibleIf":""
14     }]}]
15   };
16
17   var survey = new Survey.Model(surveyJSON);
18
19   function sendDataToServer(survey) {
20     $("#preview").append("<h5>" + "The results are:" +
           JSON.stringify(survey.data) + "</h5>");
21     $("#preview").append("<button onclick='validate()'>" + "
           Test again" + "</button>");
22   }
23 } catch (err) {
24   alert(err);
25 }
26
27 $("#preview").Survey({
28   model: survey,
29   onComplete: sendDataToServer
30 });
```

Listing 5.1: Einbindung der SurveyJs-Bibliothek

In Zeile 4 des Listings 5.1 wird ein JSON-Objekt definiert, welches in Zeile 22 zu einem Survey-Model-Objekt konvertiert wird. Über den Survey-Listener in Zeile 32 wird der aktuelle Zustand an der gewünschte Stelle in der eigenen Website angezeigt, wie die Abbildung 5.1 illustriert. Bei Abschluss des Fragebogens lassen sich die Antworten in JSON-Format, wie in Zeile 25 zu sehen ist, entweder direkt wieder an die eigene Website anbinden oder an einen externen Server schicken.

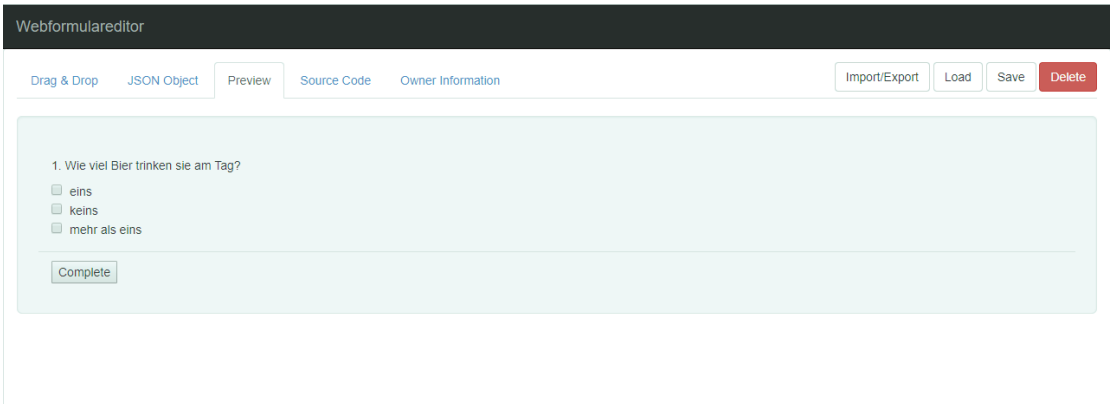

Abbildung 5.1: Einbindung des erstellten Fragebogens über die SurveyJs-Bibliothek

5.1.2 VueJs

Vue.js [9] ist ein clientseitiges JavaScript-Webframework zur Erstellung von Webanwendungen nach dem MVVM-Muster. Das Model View ViewModel (MVVM) Prinzip ist eine abgewandelte Variante des Model-View-Controller-Musters (MVC). Es dient zur Trennung von Darstellung und Logik der Benutzerschnittstelle. Über einen Datenbindungsmechanismus lassen sich einfache Änderungen dynamisch an die Website anpassen. Der Implementierungsaufwand wird dadurch drastisch reduziert, da keine separaten Controller-Klassen erforderlich sind. In folgendem Listing 5.2 wird das Vorgehen anhand eines Beispiels genauer erläutert.

```
1  <div id="app">
2    <input v-model="name">
3    <p>Hallo {{name}} !</p>
4  </div>
5
6  <script>
7    new Vue({
8      el: '#app',
9      data: {
10       name: 'Ihr Name'
11     }
12   });
13 </script>
```

Listing 5.2: Einbindung einer VueJs Renderinstanz

In Zeile 7 ff. des Listings 5.2 wird eine neue Renderinstanz mit dem Aufruf „new Vue“ an das HTML-Tag mit der Id „app“ gehängt. Das angebundene Datenobjekt beschränkt sich in diesem Beispiel auf das Attribut „name“, das zu Beginn den Wert „Ihr Name“ hat. Innerhalb des HTML-Tags mit der Id „app“ wird das angebundene Datenobjekt angezeigt (siehe Zeile 3). Die Ausgabe würde „Hallo Ihr Name“ lauten. Die hier aufgebaute Verbindung ist unidirektional, im Gegensatz zu dem Aufruf in Zeile 2 der das Datenobjekt bidirektional mit einem HTML-Input-Element verbindet. Der in diesem Input-Element stehende Wert ist sowohl Eingabe, als auch Ausgabe. Über die Eingabe eines Namens in das Input-Element, wie zum Beispiel „Peter“, überführt man die Ausgabe des Paragraphen in Zeile 3 zu: „Hallo Peter“.

5.1.3 Bootstrap

Bootstrap [12] ist ein Open-Source-Framework, welches hauptsächlich auf CSS und HTML-Elementen basiert. Über optionale JavaScript-Bibliotheken lassen sich noch dynamische Aspekte bezüglich der Darstellung einstellen. Das Framework bietet verschiedene Gestaltungsvorlagen für HTML-Elemente, wie zum Beispiel Buttons oder Tabellen.

Obwohl Bootstrap sich grundsätzlich nur mit dem visuellen Anteil einer Webanwendung auseinandersetzt, vereinfacht es dennoch den Prozess der Benutzerschnittstellenerzeugung. Die meisten Versionen von Bootstrap erlauben es Short-Cuts zu verwenden, die ganze HTML-Bereiche generieren. Es lassen sich so schnell und

unkompliziert grafische Strukturen entwickeln an die eine Renderinstanz angebunden werden kann. Mithilfe des internen Grid-Layouts von Bootstrap ist über wenige Befehle eine benutzerfreundliche Arbeitsumgebung erstellt, wie das nachfolgende Listing 5.3 zeigen soll.

```
1  <!-- TOP TAB NAVIGATION -->
2  <ul class="nav nav-tabs" role="tablist">
3    <li class="active"><a href="#tab1" role="tab" data-toggle=
       "tab">Drag & Drop</a></li>
4    <li><a href="#tab2" role="tab" data-toggle="tab">JSON
       Object</a></li>
5    <li><a href="#tab3" role="tab" data-toggle="tab" onclick="
       validate()">Preview</a></li>
6    <li><a href="#tab4" role="tab" data-toggle="tab">Source
       Code</a></li>
7    <li><a href="#tab5" role="tab" data-toggle="tab">Owner
       Information</a></li>
8  </ul>
9  <!-- TOP TAB CONTENT -->
10 <div class="tab-content">
11   <!-- DRAG AND DROP -->
12   <div class="active tab-pane fade in" id="tab1">
13   <!-- JSON OBJECT -->
14   <div class="tab-pane fade" id="tab2">
15   <!-- PREVIEW -->
16   <div class="tab-pane fade" id="tab3">
17   <!-- SOURCE CODE -->
18   <div class="tab-pane fade" id="tab4">
19   <!-- OWNER INFO -->
20   <div class="tab-pane fade" id="tab5">
21 </div>
```

Listing 5.3: Bootstrap erzeugtes Tabulatorlayout

In den Zeilen 2 bis 8 des Listings 5.3 wird eine horizontale Navigationsleiste erzeugt, welche auf die einzelnen Bereiche in Zeile 10 bis 21 referenziert. Die hier verwendete JavaScript-Bibliothek von Bootstrap identifiziert die einzelnen Bereiche über eine eindeutige Id und blendet alle nicht als „active“ markierten Bereiche aus. Die Navigationselemente aus Zeile 2 bis 8 können jeweils einen Bereich referenzieren und lassen sich über ein Anklicken in den Zustand „active“ versetzen, der im „class“-Attribute festgehalten wird.

5.2 Generische Programmierkonzepte

Bei der generischen Programmierung handelt es sich um ein Verfahren, das es erlaubt wiederverwendbare Code-Abschnitte zu entwickeln. Dabei wird die entwickelte Software möglichst allgemein gehalten, um die Anwendung auf unterschiedliche Datenstrukturen zu gewährleisten.

In den folgenden Abschnitten wird der Einsatz von Templates als generisches Programmierkonzept, sowie die Einbindung in das System, welches im Rahmen dieser Arbeit entwickelt wurde, vorgestellt. Durch die Verwendung von Templating vereinfacht sich der Prozess der Erzeugung von Fragebogenelementen, die ein wesentlichen Anteil der aufgestellten Anforderungen sind.

5.2.1 Templating

Der Begriff „Templating“ bezeichnet einen Vorgang, der es ermöglicht eine Vorlage (engl. Template) mit definierten Platzhaltern verwenden zu können. Es handelt sich hierbei um einen Programmiergrundsatz, der vor allem in der Webentwicklung Anwendung findet. Kommt ein solches Template zum Einsatz, dann werden die definierten Platzhalter durch aktuelle Inhalte ersetzt.

Im Kontext dieser Arbeit stellen die Datenobjekte, welche die Eigenschaften der Fragebogenelemente inne haben, die Übergabeparameter dar. Das Listing 5.4 zeigt ein solches Datenobjekt, welches unter dem Einsatz eines Template als Fragebogenelement in einer Webumgebung auftaucht. In der Abbildung 5.2 kann das Ergebnis dieses Vorgangs betrachtet werden.

```
1  function InputText(count) {
2    this.type = "text";
3    this.title = "";
4    this.isRequired = false;
5    this.inputType = "text";
6    this.name = "question"+count;
7    this.placeHolder = "answer";
8    this.size = 20;
9    this.width = "100%";
10   this.visibleIf = "";
11 }
```

Listing 5.4: InputText Klasse

Die Attribute, die im Listing 5.4 Zeile 2 ff. zu sehen sind, definieren die Eigenschaften eines Fragebogenelements bei der Integration in ein Template. Bei dem angegebenen Datenobjekt handelt es sich um eine einfache Texteingabe, wie sie in den funktionalen Anforderungen beschrien wurde. Durch das Attribut in Zeile 5, welches standardmäßig auf den Wert „text“ gesetzt ist, lässt sich zum Beispiel die Eingabemethode der Frage ändern. Eine veränderte Eigenschaft könnte die Eingabe von Passwörtern oder Farben zur Folge haben.

Das zugehörige Template für die Darstellung ist in Listing 5.5 zu sehen. Der Einfachheit halber wird auf das Formular-Template, welches für die Bearbeitung der Eigenschaften des Datenobjekts zuständig ist, verzichtet. Der Aufbau unterscheidet sich im Wesentlichen nur von der Möglichkeit über die Benutzeroberfläche die Eigenschaften des Datenobjekts zu verändern. Diese Eigenschaft wird über aktive Input-Elemente in der HTML-Programmierung realisiert.

```
1  var InputTextView = {
2
3    props: ['elem'],
4
5    template:
6
7    '<div>' +
8    '<h5>' +
9    '{{isTrue(elem.isRequired)}}{{elem.title || elem.name}}' +
10   '</h5>' +
11   '<input class="form-control" :size="elem.size"' +
12   ' :type="elem.inputType"' +
13   ' :placeholder="elem.placeHolder"' +
14   ' :style="{ width: elem.width}" readonly>' +
15   '</div>',
16
17   methods: {
18
19     isTrue: function (value) {
20
21       return (value ? '*' : '');
22
23     }
24   }
25 };
```

Listing 5.5: HTML-Template des InputText-Fragebogenelements

Das Template zur Darstellung des InputText-Fragebogenelements in Listing 5.5 tritt, wie das Datenobjekt selbst, als JSON-Objekt auf. In Zeile 3 wird unter dem Attribute „props“ die Variable „elem“ als Übergabeparameter festgelegt. Das eigentliche Template wird in Zeile 5 ff. in Form eines Strings definiert, welcher den HTML-Syntax imitiert. Bei dem Übergabeparameter „elem“ handelt es sich um keinen festen Datentyp. Es können beliebige Datenstrukturen übergeben werden, welche die Anforderungen des Templates erfüllen. Eine solche Anforderung ist in Zeile 12 zu sehen. Die übergebene Datenstruktur, bei der es sich in diesem Kontext um ein JSON-Objekt handelt, muss das Attribute „inputType“ besitzen, um dem HTML-Input-Element einen Typ zuzuweisen. Bei der Initialisierung eines Fragebogenelements vom Typ Texteingabe würde, aufgrund der Standardwerte des InputText-Datenobjekts, der Typ dieses HTML-Input-Elements als „text“ definiert werden.

Diese Form der Vorlage wurde in der Implementierung für jedes Fragebogenelement, das in den Anforderungen definiert wurde, umgesetzt. Dadurch lässt sich eine dynamische Einbindung der Fragebogenelemente realisieren.

5.2.2 Dynamische Einbindung von generischem Code

In diesem Abschnitt wird die Einbindung der zuvor vorgestellten Templates gezeigt. Damit die erstellten Temlpates in der darstellende HTML-Seite dynamisch verwendet werden können müssen sie zuvor an der in Kapitel 5.1.2 vorgestellte Renderinstanz angemeldet werden. Dieser Schritt wird in Listing 5.6 exemplarisch gezeigt.

```
1  var vue = new Vue({
2
3    components: {
4
5      'input-text-view': InputTextView
6
7    }
8  });
```

Listing 5.6: Einbindung von einem HTML-Template

In der Renderinstanz von Vue.js lassen sich Templates in Form von Komponenten unter dem Attribute „components“ anbinden. In Zeile 5 des Listings 5.6 wird die Variable „InputTextView“, welche das Template beinhaltet an den Tag „input-text-

view“ gebunden. So wird sicher gestellt, dass Änderungen innerhalb des Templates von der Renderinstanz registriert werden.

Der Aufruf eines Templates im HTML-Kontext ist, wie Listing 5.7 zeigt, unkompliziert. Die Verwendung von Templates erfüllt außerdem einen weiteren Vorteil, der vor allem bei der Entwicklung von Softwaresystemen dem Entwickler zu Gute kommt. Die Reduzierung des Quellcodes vereinfacht die Handhabung der Programmabschnitte und sorgt für eine klarere Struktur.

```
1  <input-text-view v-if="element.type == 'text'"
2    :elem="element">
3  </input-text-view>
```

Listing 5.7: HTML-Platzhalter für das InputText-Fragebogenelement

In Zeile 1 des Listings 5.7 wird über den Aufruf des zuvor an der Renderinstanz definierten Tags „input-text-view“ das Template aus Listing 5.5 in die HTML-Seite eingefügt. Bei dem Attribut „elem“ dieses neuen HTML-Elements in Zeile 2 handelt es sich um den Übergabeparameter der zuvor im Template festgelegt worden ist. Das im Listing 5.7 angegeben HTML-Tag tritt in der Implementierung in einem dedizierten Bereich auf, welcher über eine Drag & Drop Funktion Fragebogenelemente in Form von JSON-Objekten entgegennehmen kann. Das Fragebogenobjekt tritt hier in der Variable „element“ auf. Durch die Abfrage „v-if“ in Zeile 1 des View-Models, das in Kapitel 5.1.2 vorgestellt wurde, wird entschieden, welches Template bei der Eingabe eines Fragebogenelements gerendert wird.

Das Ergebnis des gerenderten Templates mit dem Fragebogenelement aus Listing 5.4 sieht, wie Abbildung 5.2 zeigt, aus.

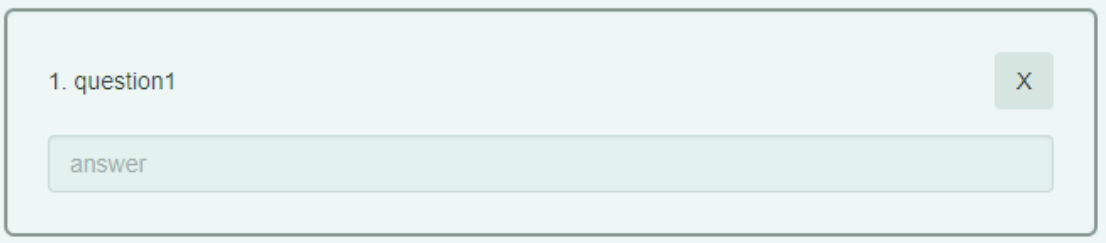

Abbildung 5.2: HTML-Ansicht eines InputText-Fragebogenelements

5.3 Benutzerschnittstelle

Die Benutzerschnittstelle eines Systems steht proportional zur Erfolgsrate einer Anwendung. Sie definiert den Bereich über den der Benutzer mit den Funktionalitäten der Software in Kontakt tritt. Die in dieser Arbeit zum Einsatz kommende Schnittstelle ist geprägt durch die Möglichkeiten eines Webbrowsers.

In den folgenden zwei Abschnitten wird auf die Darstellung des System, sowie die Funktion von Drag & Drop als Eingabemethode eingegangen.

5.3.1 Darstellungskonzept

Die Problematik der Bereitstellung von Funktionen für den Benutzer ohne ihn zu überfordern ist in der Informatik allgegenwärtig. Das KISS-Prinzip (eng.: Keep it simple, stupid) [3] dient in den meisten dieser Fälle als Leitfaden für eine erfolgreiche Softwareentwicklung.

Insbesondere in dieser Implementierung ist eine anwenderfreundliche Benutzerschnittstelle von großem Wert, wie sich aus den nicht-funktionalen Anforderungen ableiten lässt. Zur Gewährleistung einer intuitiven und einfachen Benutzeroberfläche ist es wichtig bei der Entwicklung einer Anwendungssoftware auf allgemein anerkannte Designkonventionen zurückzugreifen. Die Wahl in dieser Implementierung fiel in Folge dessen auf das Open-Source Bootstrap-Framework, welches weltweit am häufigsten in diesem Bereich zum Einsatz kommt.

Die Abbildung 5.3 zeigt die fertige Benutzeroberfläche zusammen mit einer Matrix-Multiplechoicefrage im Entwurfszustand.

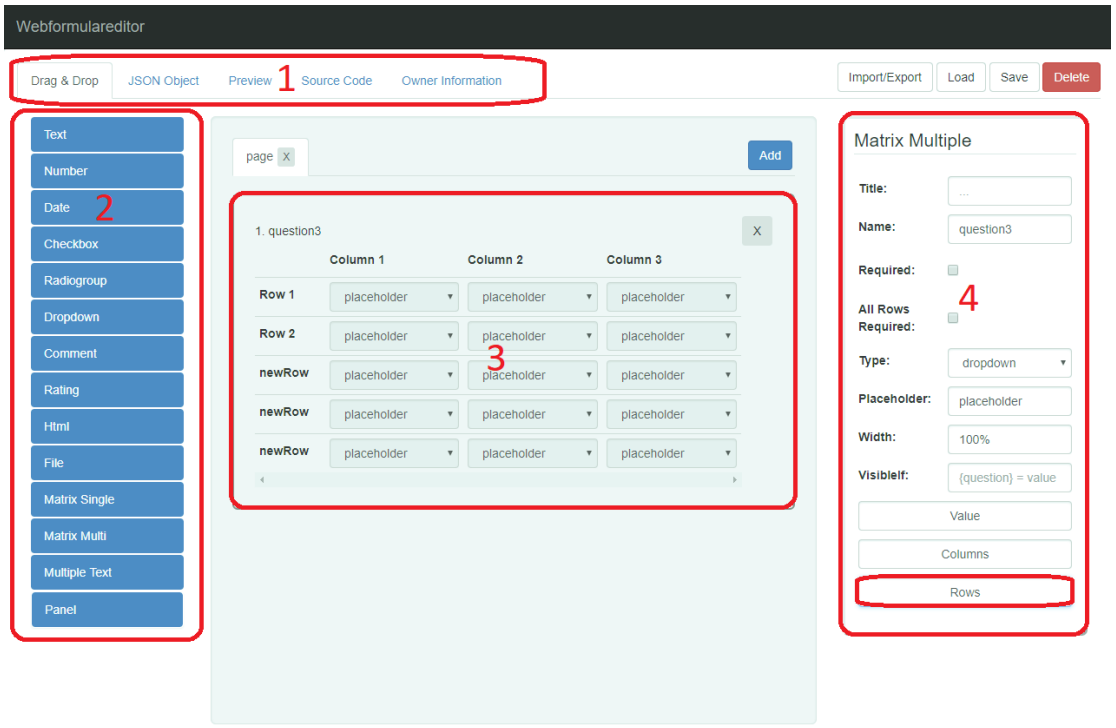

Abbildung 5.3: Ansicht einer Matrix-Multiplechoicefrage

Eine klare und verständliche Struktur sind Grundeigenschaften, die eine Anwendersoftware besitzen muss. In der Abbildung 5.3 wird dies veranschaulicht durch die markierten Bereiche. Der Benutzer hat in Bereich ① die Möglichkeit über eine Menüleiste den aktuellen Kontext zu wechseln. Durch die Trennung der Bereiche wird eine Überladung der Anwendung verhindert und der Fokus auf die einzelnen Stadien bei der Fragebogenentwicklung gerichtet. Der Bereich ② gibt dem Benutzer eine mögliche Auswahl an Fragebogenelementen zur Verfügung, die in den Bereich ③ für eine weitere Bearbeitung gezogen werden können.

Eine weitere Demonstration des eingesetzten Darstellungskonzeptes wird in der Abbildung 5.4 deutlich gemacht. Damit weiterhin eine übersichtliche Oberfläche im Sinn des KISS-Prinzip erhalten bleibt ohne den Umfang der Anwendungsfunkionen einzuschränken, verweist die Funktion der Reihenbearbeitung in Bereich ④ der Abbildung 5.3 auf ein modales Fenster, wie es in der Abbildung 5.4 zu sehen ist.

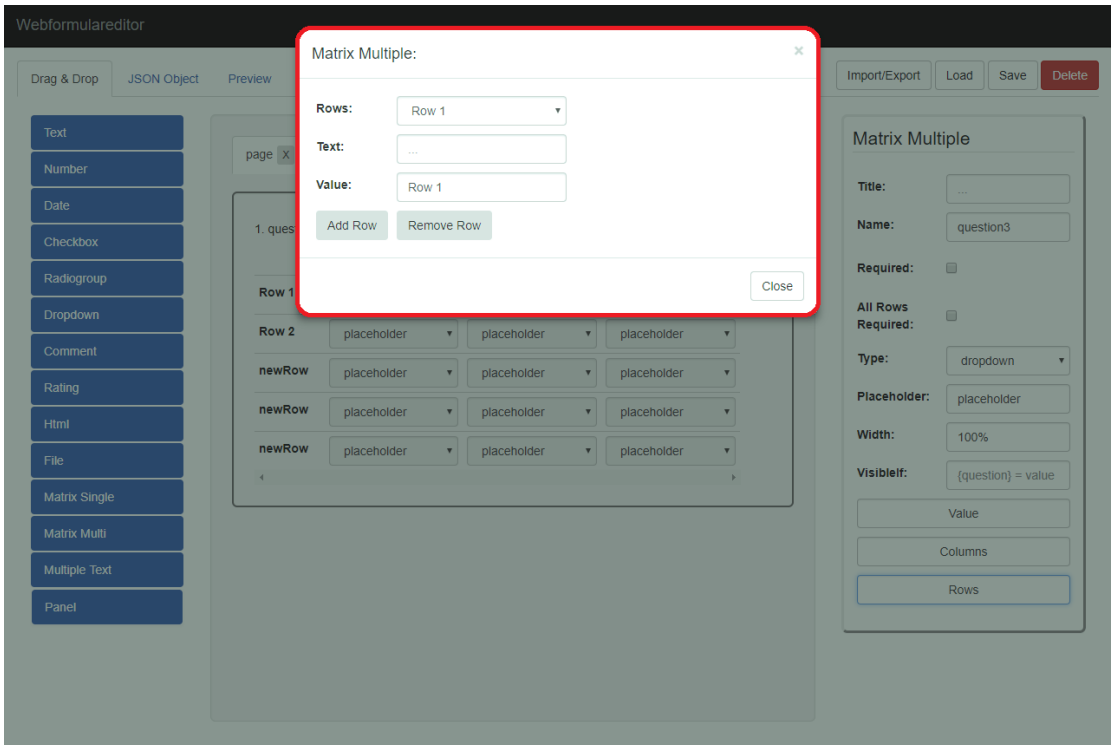

Abbildung 5.4: Bearbeitung der Reihen einer Matrix-Multiplechoicefrage

5.3.2 Drag & Drop

Einer der größten Aspekte bei der Entwicklung der Benutzerschnittstelle ist die Möglichkeit Objekte über Drag & Drop einzubinden. Diese Funktion bietet eine intuitive Bedienung von Anwendungen, die vor allem durch die allgegenwärtige Verwendung zu Stande kommt. Durch den Einsatz von Drag & Drop, der es ermöglicht Objekte mit einem Zeiger aus einem Bereich in einen Anderen zu Überführen, erweitert sich die Einsatzmöglichkeit der Anwendung auf mobile Endgeräte, wie Tablets oder Smartphones. Diese besitzen in der Regel einen Touch-Screen-Eingabe, die dem Benutzer gestattet einen oder mehrere Finger als Zeiger zu benutzen.

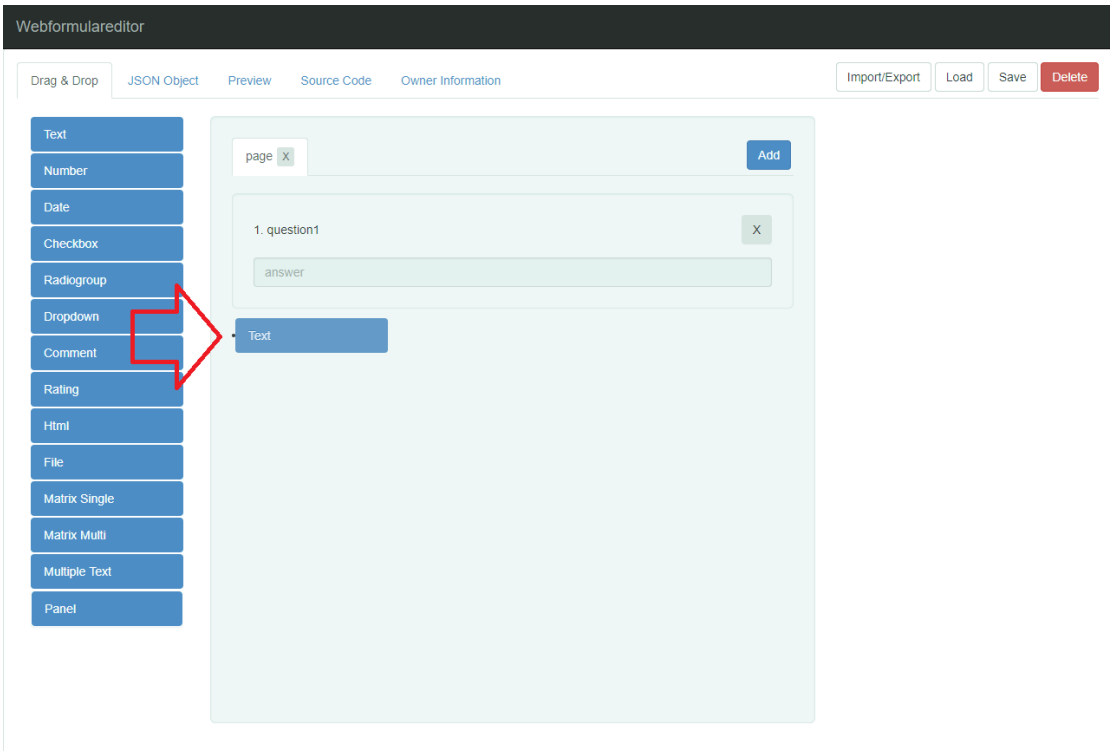

Abbildung 5.5: Einbindung eines Fragebogenelements über Drag & Drop

Die Abbildung 5.5 dient der schematische Darstellung eines Drag & Drop Vorgangs in der entwickelten Anwendungssoftware. Die in blau gezeichneten Schaltflächen repräsentieren mögliche Fragebogenelemente die zur Auswahl stehen. Über ein einfaches Anklicken und ziehen an die gewünschte Stelle können die Fragebogenelemente an den digitalen Fragebogen angeheftet werden.

In Listing 5.8 wird ein Drag-Bereich definiert der beim Laden der Anwendung durch das Iterieren über eine Liste (Zeile 2) die einzelnen Fragebogenelemente als blaue Schaltfläche, wie in Abbildung 5.5 gezeigt wird, darstellt. In Zeile 9 wird der Name des jeweiligen Fragebogenelements angegeben. Es wird der Datenbindungsmechanismus verwendet, der in Kapitel 5.1.2 vorgestellt wurde. In Zeile 4 wird definiert, welche Bezeichnung die Elemente in diesem Bereich zugewiesen bekommen und welche Aktion bei einem Drag- oder Drop-Vorgang zum Einsatz kommt. Da es sich in diesem Fall um die Auswahl an Fragebogenelementen handelt wird die „put“-Funktion ausgestellt und die „pull“-Funktion auf das Attribute „clone“ gesetzt. Die Standardeinstellung für den „clone“-Aufruf bewirkt eine exakte Kopie des ausgewählten Elements und erlaubt somit bei einer Wiederholung dieses Vorgangs keine eindeutige Zuordnung der Fragebogenelemente. In Folge dessen wird in Zei-

le 5 diese Funktion überschrieben mit einem Aufruf der ein neues Objekt aus einer der Fragebogenklassen, wie in Listing 5.4, erstellt.

```
1  <draggable v-model="elements"
2    v-for="(element, index) in elements"
3    class="nav nav-pills nav-stacked dragArea"
4    :options="{group:{ name:'elements',  pull:'clone',
       put:false }, sort:false}"
5    :clone="(original) => {element.name != 'Panel' ?
       json.meta.counter += 1 : 0; return new Choice(
       element.name, json.meta.counter); }"
6    style="padding: 1px">
7
8    <li class="active" :class="{ panel: isPanel(element.name)
       }">
9      <a>{{element.name}}</a>
10   </li>
11 </draggable>
```

Listing 5.8: Drag-Bereich der Anwendung

Das nachfolgende Listing 5.9 beschreibt den Drop-Bereich der Anwendung. Im Gegensatz zum Drag-Bereich in Listing 5.8 können hier Elemente vom Typ „elements“ und „panels“ eingebunden werden, wie Zeile 4 zeigt. Das „pull“-Attribut ist in diesem Listing nicht aufgeführt und dadurch standardmäßig auf „true“ gesetzt. Über die Möglichkeit in diesem Bereich Drag & Drop zu verwenden lassen sich bereits eingebundene Fragebogenelemente erneut in ihrer Reihenfolge sortieren.

```
1  <draggable
2    v-model="page.elements"
3    class="dragArea"
4    :options="{group:{name:'list', put:['elements', 'panels
       ']}, animation: 100}"
5    style="min-height: 100px">
6  </draggable>
```

Listing 5.9: Drop-Bereich der Anwendung

6 Anforderungsabgleich

In diesem Kapitel werden die zuvor aufgestellten Anforderungen aus Kapitel 3.2 mit der Umsetzung in der Implementierung verglichen. Es wird wieder zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden.

6.1 Funktionale Anforderungen

Die zuvor definierten funktionalen Anforderungen aus Kapitel 3.2.1 werden in diesem Abschnitt auf ihren Erfüllungsgrad geprüft. Dafür wird eine Skala verwendet, die den aktuellen Zustand der Anforderung in der Implementierung beschreibt. Die Skala reicht von **(4)** Anforderung vollständig bis **(0)** nicht erfüllt. Diese Kategorisierung dient zur Analyse der Umsetzung des Systems. Anhand der gegebenen Werte lässt sich qualitativ feststellen, wie erfolgreich die Umsetzung und somit die Lösung für die angeführten Anforderungen ist.

(4) Die Anforderung wurde vollständig erfüllt

(3) Die Anforderung wurde ausreichend erfüllt

(2) Die Anforderung wurde unzureichend erfüllt

(1) Die Anforderung wurde konzipiert, aber nicht implementiert

(0) Die Anforderung wurde nicht erfüllt

BEZEICHNUNG	PRIORITÄT	ZUSTAND
Speichern	*MUSS*	**(4)**
Laden	*MUSS*	**(4)**
Löschen	*MUSS*	**(4)**

Externe Anbindung	*MUSS*	**(4)**
Import	*SOLL*	**(4)**
Export	*SOLL*	**(4)**
Drag & Drop	*SOLL*	**(4)**
Preview	*SOLL*	**(4)**
Direkte Eingabe	*SOLL*	**(4)**
Metadaten	*SOLL*	**(4)**
Validierung	*SPÄTER*	**(0)**

Umsetzung analoger Fragebogenelemente

Texteingabe	*MUSS*	**(4)**
Zahleneingabe	*MUSS*	**(4)**
Zeit-/Datumseingabe	*MUSS*	**(4)**
Multiplechoicefragen	*MUSS*	**(4)**
Gewichtungseingaben	*MUSS*	**(4)**
Anmerkungen	*MUSS*	**(4)**
Informationstexte	*MUSS*	**(3)**
Gruppierung	*SOLL*	**(4)**
Seitenumbruch	*SOLL*	**(4)**
Bilderanzeige	*SOLL*	**(2)**

Erweiterung der Fragebogenelemente

Filterfragen	*MUSS*	**(3)**
Dropdown-Auswahl	*MUSS*	**(4)**
Matrix-Auswahl	*MUSS*	**(4)**
Dateiupload	*SOLL*	**(4)**
Bewegte Bilderanzeige	*KANN*	**(2)**

6.2 Nicht-funktionale Anforderungen

In diesem Abschnitt wird anhand der Implementierung der Zustand, der in Kapitel 3.2.1 definierten nicht-funktionalen Anforderung der Erfüllungsgrad untersucht. Zur Kategorisierung wird folgende Skala verwendet:

(4) Die Anforderung wurde vollständig erfüllt

(3) Die Anforderung wurde ausreichend erfüllt

(2) Die Anforderung wurde unzureichend erfüllt

(1) Die Anforderung wurde konzipiert, aber nicht implementiert

(0) Die Anforderung wurde nicht erfüllt

Im Gegensatz zur Analyse der funktionalen Anforderungen, gibt diese Kategorisierung Aufschluss darüber, wie gelungen das System insgesamt ist. Unabhängig davon, wie erfolgreich die Umsetzung der funktionalen Anforderungen ist, führt zum Beispiel eine schlechte Benutzbarkeit oder Erreichbarkeit unweigerlich immer zum Misserfolg der Anwendung.

BEZEICHNUNG	**PRIORITÄT**	**ZUSTAND**
Benutzbarkeit	*MUSS*	**(4)**
Zuverlässigkeit	*MUSS*	**(3)**
Portierbarkeit	*MUSS*	**(4)**
Korrektheit	*MUSS*	**(2)**
Aussehen und Handhabung	*SOLL*	**(4)**
Leistung und Effizienz	*SOLL*	**(4)**
Sicherheitsanforderungen	*SOLL*	**(4)**
Flexibilität	*KANN*	**(4)**
Erweiterbarkeit	*KANN*	**(1)**

7 Zusammenfassung und Ausblick

In diesem Kapitel wird die Arbeit zusammenfassend noch einmal aufgeführt. Es werden die wesentlichen Erkenntnisse und die Zielsetzung dieser Arbeit aufgegriffenen, um einen Ausblick auf zukünftige Verwendungsmöglichkeiten zu geben.

7.1 Zusammenfassung

Diese Arbeit hat sich mit dem Problem der fehlenden Möglichkeiten für eine digitalisierte Vorgehensweise bei der Entwicklung von Fragebögen auseinandergesetzt. Dazu musste zuerst festgestellt werden, welche Gründe es für die Beibehaltung der herkömmlichen papiergebundenen Erstellung gibt. Der Fokus lag dabei auf dem Bereich der Psychologie, da diese Benutzergruppe durch eine Lösung stark profitiert.

Einer der Gründe für die Verwendung von einer papiergebundenen Entwicklung ist die fehlende IT-Kenntnis von Psychologen, die benötigt wird um lauffähige Software zu entwickeln. Ein weiterer Grund ist der immense Kostenaufwand bei einer Entwicklung von softwaregestützten Systemen. Darüber hinaus garantiert eine extern entwickelte Software keinen hundertprozentigen Erfolg. Das wiederum liegt vor allem an der fehlenden Fachkenntnis der Softwareentwickler. In den wenigsten aller Fälle ist der entwickelnde Softwarespezialist ebenfalls ein Fachexperte der Psychologie, was zu Fehlinterpretationen von Anforderungen in der Entwicklung der Software führen kann.

Diese Arbeit hat in Folge dessen Anforderungen auf der Grundlage gängiger Fragebögen definiert, die es ermöglichen einen Fragebogen auf digitalem Wege zu entwickeln. Es wurden relevante Stellen des Entwicklungsprozesses eines Fragebogens betrachtet, um eine konkrete Vorstellung des Entwicklungsprozesses zu erlangen.

Das Resultat dieser Arbeit ist ein webbasierter Editor zur Erstellung von Fragebögen mit einfacher Handhabung, der als Werkzeug dienen soll, um die fehlenden

Fachkenntnisse der beiden Bereiche, Software-Entwicklung und Psychologie, auszugleichen.

7.2 Ausblick

Wie aus dem Anforderungsabgleich hervorgeht sind die meisten Anforderungen in der Implementierung vollständig umgesetzt worden. Jedoch stellte sich heraus, dass die Validierung des Datenobjekts einen größeren Umfang hat, als ursprünglich geplant war. Die Konzeption und Entwicklung eines solchen Systems lässt sich weiter mit einer automatisierten Auswertung koppeln, die der nächste Schritt bei der Weiterführung des Projektes ist.

Eine automatisierte Auswertung des resultierenden Datenobjekts mit vorhergehender Validierung führt zu einer weiteren Zeitersparnis bei der Fragebogenentwicklung. Für eine große Flexibilität muss das System, welches nach Vorgaben des Anwenders die Resultate einer Umfrage selektiert und grafisch darstellt, als unabhängige Erweiterung entwickelt werden. Das garantiert eine Benutzung zur Validierung und Auswertung außerhalb der in dieser Arbeit entwickelten Anwendung. Diese Erweiterung stellt nur eine Möglichkeit dar, wie das entwickelte System ergänzt werden könnte.

Eine weitere Anpassung im Rahmen dieser Arbeit ist die komfortablere Einbindung von Bildern und Videos in den Fragebogen. Eine Schnittstelle für benutzerdefinierte Fragebogenelemente ist ebenso denkbar. Aufgrund der Import-/Export-Funktion und der einfach gehaltenen Architektur bietet die entstandenen Implementierung eine solide Grundlage für weitere Projekte.

Literaturverzeichnis

[1] *Fragebogen*. https://de.wikipedia.org/wiki/Fragebogen, Abruf: 20.10.2017

[2] *JavaScript HTML DOM*. https://www.w3schools.com/js/js_htmldom.asp, Abruf: 26.11.2017

[3] *KISS-Prinzip*. https://de.wikipedia.org/wiki/KISS-Prinzip, Abruf: 01.12.2017

[4] *LimeSurvey: the online survey tool - open source surveys*. https://www.limesurvey.org/de/, Abruf: 26.11.2017

[5] *MobileTx*. https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/mobiletx/, Abruf: 05.12.2017

[6] *MoSCoW-Priorisierung*. https://de.wikipedia.org/wiki/MoSCoW-Priorisierung, Abruf: 21.11.2017

[7] *Project SurveyJS: Open-Source JavaScript & Web API Survey Management*. https://surveyjs.io/, Abruf: 20.11.2017

[8] *QuestionSys - A Generic and Flexible Questionnaire System Enabling Process-Driven Mobile Data Collection*. https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/questionsys/, Abruf: 02.11.2017

[9] *Vue.js*. https://vuejs.org/, Abruf: 20.11.2017

[10] *Fragetyp - Matrix (Ja/Nein/Unsicher)*. https://manual.limesurvey.org/Question_type_-_Array_(Yes-No-Uncertain)/de. Version: März 2013, Abruf: 26.11.2017

[11] *Die digitale Gesundheitswirtschaft: Potenziale für die MedTech-Branche*. https://www.bvmed.de/de/technologien/trends/die-digitale-gesundheitswirtschaft-potenziale-fuer-die-medtech-branche. Version: Mai 2016, Abruf: 18.11.2017

[12] EFRON, B. ; TIBSHIRANI, R. J.: *An Introduction to the Bootstrap*. Boca Raton, Florida, USA : Chapman & Hall/CRC, 1993 (Monographs on Statistics and Applied Probability 57)

[13] KREMPL, S. : *Elektronische Steuererklärung gewinnt langsam an Akzeptanz*. https://www.heise.de/newsticker/meldung/Elektronische-Steuererklaerung-gewinnt-langsam-an-Akzeptanz-188758.html. Version: Juli 2008, Abruf: 05.11.2017

[14] PROBST, T. ; PRYSS, R. ; LANGGUTH, B. ; SPILIOPOULOU, M. ; LANDGREBE, M. ; VESALA, M. ; HARRISON, S. ; SCHOBEL, J. ; REICHERT, M. ; STACH, M. ; SCHLEE, W. : Outpatient Tinnitus Clinic, Self-Help Web Platform, or Mobile Application to Recruit Tinnitus Study Samples? In: *Frontiers in Aging Neuroscience* 9 (2017), April, S. 113–113

[15] RICHTERS, K. : *Die digitale Anamnese*. https://www.gruenderszene.de/allgemein/heartbeat-operation-portraet. Version: Juni 2016, Abruf: 03.11.2017

[16] SCHICKLER, M. ; PRYSS, R. ; STACH, M. ; SCHOBEL, J. ; SCHLEE, W. ; PROBST, T. ; LANGGUTH, B. ; REICHERT, M. : An IT Platform Enabling Remote Therapeutic Interventions. In: *30th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2017)*, IEEE Computer Society Press, Juni 2017

[17] SCHOBEL, J. ; PRYSS, R. ; REICHERT, M. : Using Smart Mobile Devices for Collecting Structured Data in Clinical Trials: Results From a Large-Scale Case Study. In: *28th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2015)*, IEEE Computer Society Press, Juni 2015, S. 13–18

[18] SCHOBEL, J. ; PRYSS, R. ; SCHICKLER, M. ; RUF-LEUSCHNER, M. ; ELBERT, T. ; REICHERT, M. : End-user programming of mobile services: empowering domain experts to implement mobile data collection applications. In: *Mobile Services (MS), 2016 IEEE International Conference on* IEEE, 2016, S. 1–8

[19] SCHOBEL, J. ; PRYSS, R. ; SCHLEE, W. ; PROBST, T. ; GEBHARDT, D. ; SCHICKLER, M. ; REICHERT, M. : Development of Mobile Data Collection Applications by Domain Experts: Experimental Results from a Usability Study. In: *29th International Conference on Advanced Information Systems Engineering (CAiSE 2017)*, Springer, Juni 2017 (LNCS 10253), S. 60–75

[20] SCHOBEL, J. ; SCHICKLER, M. ; PRYSS, R. ; MAIER, F. ; REICHERT, M. : Towards Process-Driven Mobile Data Collection Applications: Requirements, Challenges, Lessons Learned. In: *10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps*, 2014, S. 371–382

[21] SOBIESKI, A. : *Digital forms, questionnaires, surveys and opinion polls*. `https://www.w3.org/community/egovernance/2013/08/22/digital-forms-questionnaires-surveys-and-opinion-polls/`. Version: August 2013, Abruf: 23.10.2017

Abbildungsverzeichnis

Name: Maximilian Scheurich Matrikelnummer: 857517

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Maximilian Scheurich